Shubham Joshi 437962

Rudraksh Tyagi 455851

Husnain Mustafa 454398

**WEB SCRAPING PROJECT**

**Introduction:**

The field of education is witnessing a digital revolution with the advent of data-enabled and technology-driven platforms. In this context, the present project aims to explore and analyze the website "Careers360," an educational products and services company. Careers360 is an innovative platform that seamlessly integrates vast amounts of student and institutional data with user-generated preferences. With over 15 million monthly visitors, Careers360 leverages this data integration to develop sophisticated prediction and recommendation products for students, enabling them to explore and achieve their career plans based on their interests and abilities.

**Description of the Project:**

The primary objective of this project is to scrape specific information from the Careers360 website and perform a comparative analysis of three different scraping techniques: Beautiful Soup, Scrapy, and Selenium**. The targeted data for scraping includes college names, college degrees, sectors, and the respective URLs of colleges**. By employing these scraping methods, we aim to gather a minimum of 100 different links containing the desired information. We will use 3 different scraper mechanism .

Beautiful Soup, a Python library, provides a straightforward yet powerful approach for web scraping. Its intuitive API allows us to parse and extract information from HTML and XML documents. On the other hand, Scrapy, a robust and highly extensible web scraping framework, offers a more structured and scalable approach for data extraction. Lastly, Selenium, a browser automation tool, enables us to scrape dynamic websites that heavily rely on JavaScript for rendering content.

Upon collecting the desired data using these scraping techniques, we will proceed with a simple analysis of the gathered information. This analysis will provide insights into the distribution of colleges across various sectors and their corresponding degree offerings. By comparing the performance of the three scraping methods, we can evaluate their efficiency in terms of speed, ease of implementation, and reliability. Such a comparative analysis will help determine the most suitable approach for future web scraping projects, considering the specific requirements and challenges involved.

**Description of scraper mechanics**

Beautiful Soup

The code utilizes Beautiful Soup and lxml in conjunction to parse and extract specific elements from the HTML content of each page. It then collects the desired data (college names, Degree , sectors, and the URL ) and stores it in lists. Finally, the scraped data is organized into a dictionary and saved as a CSV file for further analysis.

The code is a scraper implemented using Beautiful Soup to extract data from the Careers360 website. The code begins by importing the necessary libraries, including Beautiful Soup, requests, lxml, time, and pandas.

The code then sets a Boolean parameter to determine whether to limit the scraping to a specific number of pages or scrape up to a maximum of 100 pages. It initializes empty lists to store the scraped data.

A loop is used to iterate through the desired number of pages. For each page, a GET request is made to fetch the HTML content. Beautiful Soup is used to parse the HTML and extract specific elements such as college names, ranks, sectors, and user ratings using XPath expressions.

The extracted data is appended to their respective lists. After scraping the desired data from all the pages, a dictionary is created to store the scraped data. The data is then saved in a CSV file using pandas DataFrame.

**Selenium**

The code initializes the Selenium web driver and sets up the necessary configurations. It opens the target website and scrapes data from multiple pages. The data includes school names, courses, sectors, and links. The scraped data is stored in lists, organized into a dictionary, and saved as a CSV file. Finally, the driver instance and browser window are closed.

The code initializes lists to store the scraped data. It then iterates over the desired number of pages, using the driver to locate and extract specific elements such as school names, courses, sectors, and links. The data is appended to their respective lists.

**Scrapy**

The code utilizes the Scrapy framework, which is specifically designed for web scraping. The Scrapy code navigates through the specified pages, extracts relevant information using XPath expressions, converts the data into a structured format, and stores it in a CSV file. This process is automated using the Scrapy framework, making it efficient and reliable for web scraping tasks.

**Technical description of the output**

The output consists of a tabular format with four columns: Name, Degree, Sector, and URL. Each row represents a different college and provides information about the college's name, degree or program offered, the sector (private, government, etc.) to which the college belongs, and a URL that directs to the respective college's page on the Careers360 website.

This tabular output serves as a structured representation of the scraped data obtained from the website. It allows for easy comprehension and analysis of the information, enabling further exploration and manipulation of the data as needed.

**Data Analysis**

* The data shows that total scrapped records are 5000.
* The data shows that out of the total scrapped records of 5000, the majority of the universities (4026) are private, while a smaller number (974) are public or government universities. Therefore, approximately **80.52% of the scraped data corresponds to private universities**, while around 19.48% represents public universities.
* The Below graph highlights the top 10 degrees and the corresponding number of colleges offering those degrees. It shows that B.Ed has the highest number of colleges (753), followed by B.Com (433) and MBA (314). BA, B.Pharma, B.Tech Computer Science and Engineering, Diploma in Civil Engineering, B.Tech Civil Engineering, B.Sc Nursing, and MBBS degrees also have a significant number of colleges associated with them.
* There are 542 distinct courses.

**Detailed Description of each scraper by everyone**

**Beautiful Soup:**

This part is built by Rudraksh Tyagi and The provided code is a scraper implemented using Beautiful Soup to extract data from the Careers360 website. The code begins by importing the necessary libraries, including Beautiful Soup, requests, lxml, time, and pandas.

The code then sets a boolean parameter to determine whether to limit the scraping to a specific number of pages or scrape up to a maximum of 100 pages. It initializes empty lists to store the scraped data.

A loop is used to iterate through the desired number of pages. For each page, a GET request is made to fetch the HTML content. Beautiful Soup is used to parse the HTML and extract specific elements such as college names, ranks, sectors, and user ratings using XPath expressions.

The extracted data is appended to their respective lists. After scraping the desired data from all the pages, a dictionary is created to store the scraped data. The data is then saved in a CSV file using pandas DataFrame.

Overall, the code demonstrates how Beautiful Soup can be used to scrape data from a website by parsing the HTML structure and extracting specific information of interest.

**Selenium:**  
This part of is built by Husnain Mustafa .The code uses the Selenium library, which is a powerful tool for web scraping, to extract data from a website. It begins by setting up the web driver to use the Firefox browser. Then, it specifies the URL of the website that needs to be scraped. Once the website is loaded, the code initializes empty lists to store the scraped data.

To extract the required information, the code loops through the desired number of pages. Within each iteration, it utilizes the web driver to locate specific elements on the page, such as school names, courses, sectors, and links. The code then extracts the text or attribute values from these elements and appends them to their respective lists.

After scraping the data from each page, the code moves on to the next page using the driver. This process continues until all the pages have been scraped. Once all the data has been collected, the code organizes it into a dictionary structure.

To facilitate further analysis, the code converts the dictionary into a pandas DataFrame, a tabular data structure. It then saves the DataFrame as a CSV file for easy access and manipulation.

Finally, the code closes the web driver and the browser window, completing the scraping process.

In summary, the code effectively utilizes Selenium to automate the process of navigating through the website, extracting specific information from each page, and storing it in a structured format for further analysis and manipulation.

**Scrapy:**

This part is built by Shubham Joshi .The Scrapy code provided is used to scrape data from the website careers360.com. The main goal is to extract information such as school names, courses, sectors, and links from multiple pages. The code utilizes the Scrapy framework, which is specifically designed for web scraping.

The code starts by setting up the necessary configurations, including the number of pages to scrape and the CSV file to store the data. It defines a Spider class named 'careerSpider', which inherits from the Scrapy Spider class. Within this class, the start\_urls are initialized with the URLs of the pages to be scraped.

The parse() method is responsible for extracting the desired data from the web page. It uses XPath expressions to locate specific elements containing the required information. The extracted data is then stored in lists and converted into a dictionary. The dictionary is then converted into a pandas DataFrame, which is appended to the CSV file.

Finally, the process is initiated using the CrawlerProcess class, and the crawl() method is called to start the spider. The data is scraped, processed, and saved to the CSV file. The resulting CSV file is then sorted alphabetically by school name for better organization and saved again.

In summary, the Scrapy code navigates through the specified pages, extracts relevant information using XPath expressions, converts the data into a structured format, and stores it in a CSV file. This process is automated using the Scrapy framework, making it efficient and reliable for web scraping tasks.

**Conclusion**